REMARKS

With respect to claim 1, one issue is whether the reference teaches "granting the resource to the thread of instructions." This must be done "when the resource becomes available." Thus, the observation that there is no point of time specified in the claim (see office action, page 9, paragraph 31) seems to be unsupportable. The resource must be granted <u>when</u> the resource becomes available. The argument that "... the claim requires a granting to be performed at some point in time" (office action at page 9, paragraph 31) recognizes that the granting is <u>not</u> done, at least at the time when the resource becomes available.

This is important because it is not necessary for the thread then to again request access which, with enough instances, wastes a lot of time.

Instead, the opposite is what happens in the cited reference. Once it is unlocked, the fact that the thread is unlocked is broadcast to all threads. See Figure 5, block 508 and column 8, lines 33-48. There is no granting any thread access to the variable after it has been unlocked. Clearly, all that happens is the availability of the thread is broadcast and then the threads must again request access to the resource. This is different than granting the resource to the thread when the resource becomes available.

The issue of allowing one or more or all of the threads access (paragraph 31 of the office action) misses the point. There is no granting of any access in Duval. Instead, there is merely a broadcast of availability. A thread must still request the resource and if it does not get it, wait in line. There is no automatic or granting when the resource becomes available.

Therefore, the operation is significantly different and reconsideration would be appropriate.

Similarly, claim 15 calls for automatically changing the thread to the active state and granting the resource to the thread of instructions. Again, we have the automatic granting of the resource, not simply the broadcasting of resource availability, to use the Examiner's language. Therefore, the cited reference fails to meet the claimed limitations.

The argument that the threads are allowed to continue execution (i.e. allowed to access the variables) is not the same as what is in the claim. All the threads can now attempt to access the variables. There is no automatic granting of any thread when the variable becomes available. The suggestion that, because the threads are prevented from using the resource, the granting step occurs

in order for the threads to access the resource, would mean that since all the threads receive the broadcasts of the availability, then they all have been granted access to it -- an impossibility.

Therefore, reconsideration of the rejection of claim 15 should also be reconsidered.

Continued deferral of the double patenting rejection is respectfully requested.

Respectfully submitted,

Date: <u>May 12, 2009</u>
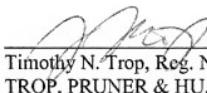
_____
Timothy N. Trop, Reg. No. 28,994
TROP, PRUNER & HU, P.C.
1616 South Voss Road, Suite 750
Houston, TX 77057-2631
713/468-8880 [Phone]
713/468-8883 [Fax]

Attorneys for Intel Corporation